面向 Coding Agent 的 Agentic RL 扩展

任务合成与环境构造

AgenticAICon 2026 · Coding Agent 技术研讨会

焦政博

香港科技大学博0/ 腾讯混元青云计划

2026年7月2日

报告路线图

01

背景

Agentic RL 正在成为主流，核心瓶颈是数据枯竭

02

Act 1：任务合成

从静态 Pipeline 到前沿自适应的进化

03

Act 2：奖励对齐

从"难度"走向"有效性"

04

Act 3：环境构造

让训练 Loop 跑得起来

05

收尾与展望

Socratic-SWE 三层合一与后续方向

Agentic RL 正在成为主流

RL 算法演进

- o1 → DeepSeek-R1：推理 RL 规模化
- GRPO / DAPO / GSPO：开源算法涌现
- SWE-RL / SWE-Gym：SWE 训练兴起
- GiGPO / ARPO / SkyRL-Agent：Agentic RL

Coding Agent 的独特挑战

- Long-horizon 多步交互
- 环境交互频繁（定位→编辑→执行→验证）
- 反馈延迟且 partial correctness
- 不是简单的 pass/fail

核心问题：如何用 RL 训练 Coding Agent?

Single-turn 推理 RL 已在数学和代码生成上取得成功，但 Coding Agent 需要在真实代码仓库中完成完整闭环，反馈是**延迟的、稀疏的**，且往往不是 binary 的。

核心瓶颈：高质量训练数据枯竭

所有方法面对同一个根本瓶颈 —— 高质量训练数据的枯竭

三个子问题

难度分布固化

模型变强后数据太简单，有效梯度消失

任务形式同质化

模板化合成数据缺乏多样性

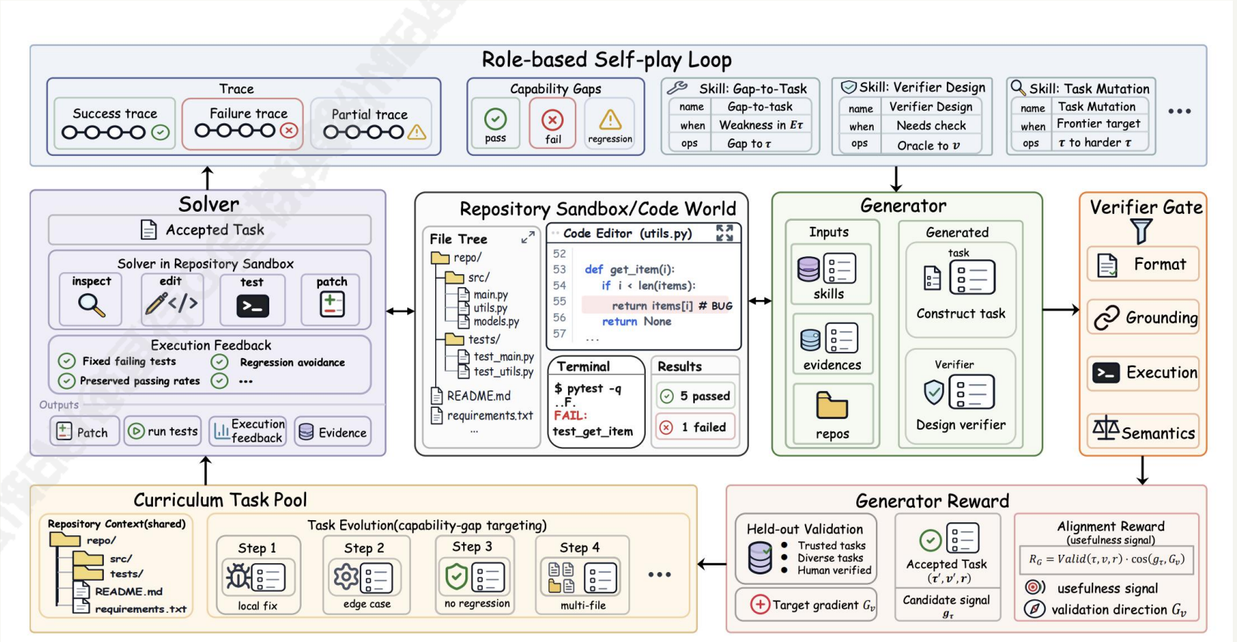
验证机制不充分

难以判断任务是否真正有用

解决方案不是"造更多数据"，而是让训练系统自己进化

需要一个能够随 Agent 能力共同成长的动态训练系统

Socratic-SWE 的闭环框架



三个层面

● Act 1

Task Synthesis

任务合成

● Act 2

Reward / Selection

奖励对齐

● Act 3

Environment

环境构造

01

ACT 1

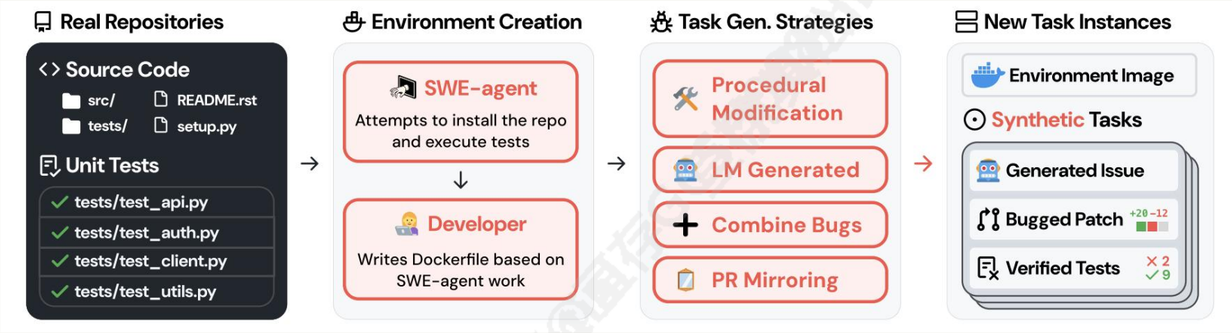
任务合成的进化

从静态 Pipeline 到前沿自适应

Task Synthesis 从 preprocessing step 变成 **trainable policy**

静态 Pipeline 的天花板 — SWE-smith

Pipeline 流程



关键数据

- **128** 个 GitHub repo → **50k** 个 task instance
- NeurIPS 2025 **Spotlight**
- AST mutation / LM-guided / PR 回退

核心局限

分布固定

模型训练三轮后，大部分 task 要么太简单要么太难，能提供有效梯度的 task 越来越少

→ 学习信号衰减，模型 saturate

根本问题：静态分布无法随模型能力动态调整

需要让 task 分布跟着模型走

Self-play：让 Task 动起来

SSR (Meta)

同一 LLM 交替扮演 inject + repair 两个角色

无需人工标注

+10.4 on Verified

R-Zero

Challenger-Solver 对抗

数学推理 self-evolving

ICLR 2026

Absolute Zero

完全 zero data

Code executor 做唯一验证信号

NeurIPS 2025

SPICE

文档语料 grounding

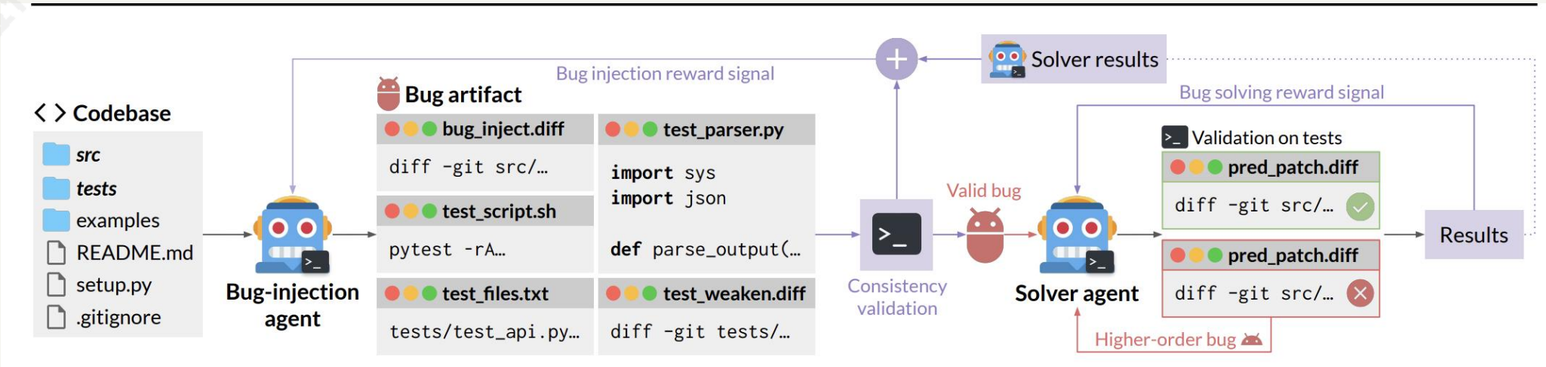
信息不对称制造 genuine challenge

进展

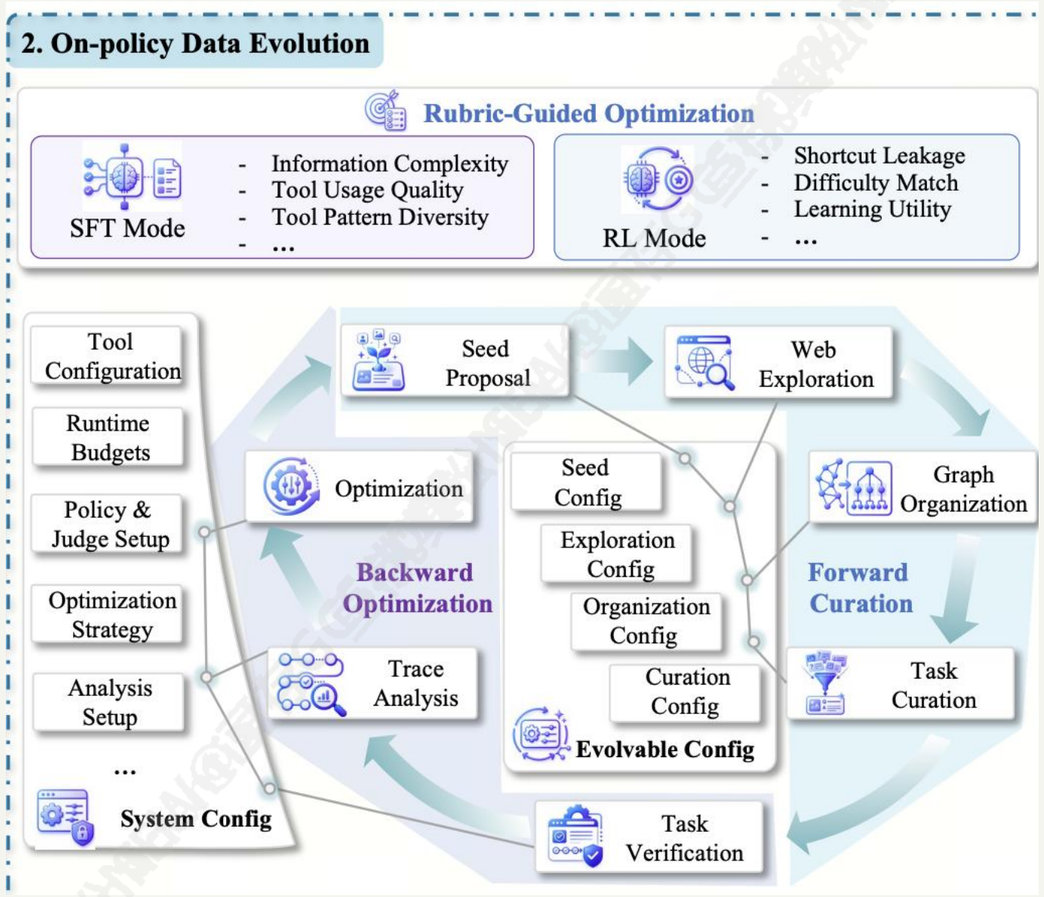
Self-play 摆脱了人工数据依赖，让 task 分布可以随模型动态演化

新问题

生成的 task 很多，但**缺乏针对性**：模型的 frontier 到底在哪？



Open-Ended 突破 — ODE



关键创新

- Judge **动态生成 rubrics** 评估 Solver
- Rubrics 本身也在**进化**
- 突破 verifiable reward 限制

对 Coding Agent 的启发

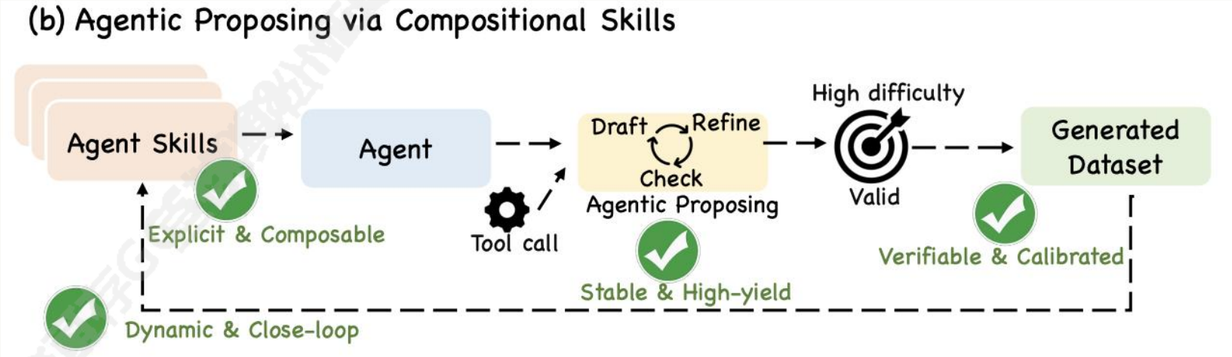
SWE 任务中"什么是好的修复"本身就不是 binary 的

→ **Partial correctness 也能做 RL**

需要更精细的评估机制替代简单的 pass/fail

技能引导 — Agentic Proposing

Goal-Driven 决策过程



关键数据

- **4B** Proposer / **11k** 数据 → 30B solver
- AIME25 **91.6%** SOTA (ICML'26)

核心信息

每条数据**精准命中**模型能力前沿

少量高质量 > 海量低质量

ICML 2026

训练方法

Multi-Granularity Policy Optimization — 多粒度策略优化训练 Proposer

Frontier Targeting — PROPEL

用 Activation Probe 替代昂贵 Rollout

传统：Solver-in-the-loop

每次 frontier targeting 需要 rollout solver

SWE 场景一次 rollout 几十分钟

Cost 不可接受

PROPEL：Probe-amortized

轻量 activation probe

一次 forward pass 预测 solver pass rate

Cost 降一个数量级

关键数据

SWE 场景 frontier task 比例：

9.8% → 19.6%

EvoTrainer

进化对象从"task"扩展到

training harness 本身也在与 policy 共同进化

核心思路：用 cheap proxy 替代 expensive rollout，实现高效的 frontier targeting

Act 1 小结

任务合成的进化线

关键转变

Task synthesis 不再是固定的 **preprocessing step**

而是 **可训练的、自适应的策略**

02

ACT 2

奖励对齐

从"难度"走向"有效性"

Hard ≠ Useful，需要 **gradient-aligned selection**

问题：Hard ≠ Useful

传统 Difficulty Proxy

Hardness reward (1−p)：越难越好

→ 极难 task 全 fail，**零梯度**

Variance reward (p≈ 0.5)：targeting frontier

→ "难在无关地方"，**无迁移价值**

SWE 场景具体例子

- 依赖 obscure C-extension 边界情况

→ 即使做对，学到的东西**不会迁移**

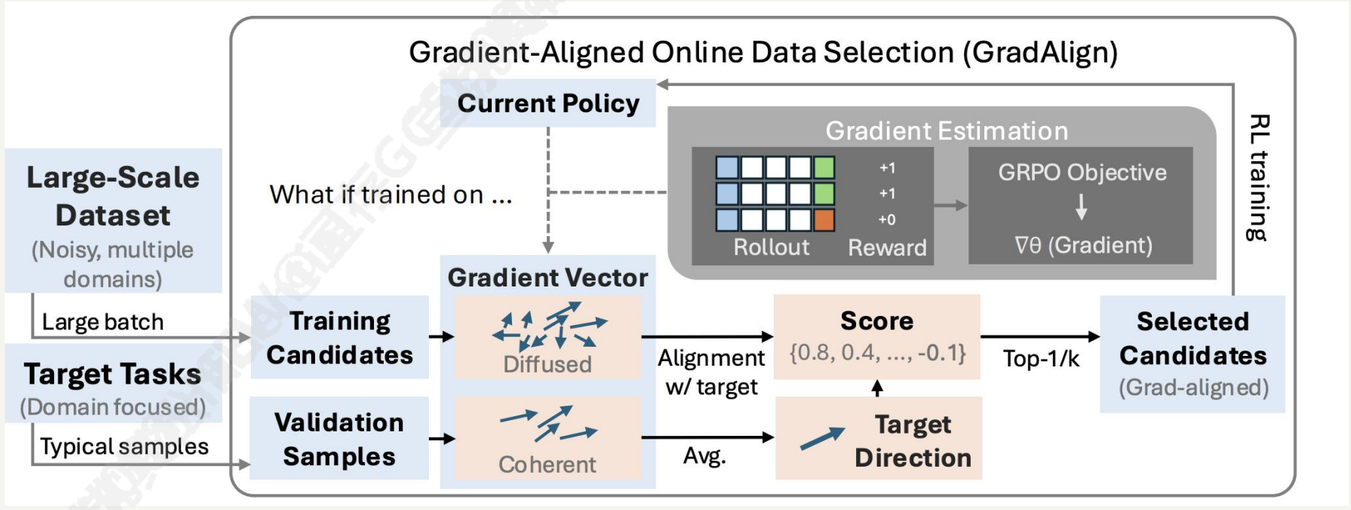
- 涉及 off-by-one 错误

→ 通用修复模式，**有迁移价值**

核心矛盾：难度和学习价值之间不存在可靠的对应关系

GradAlign — 梯度对齐选数据

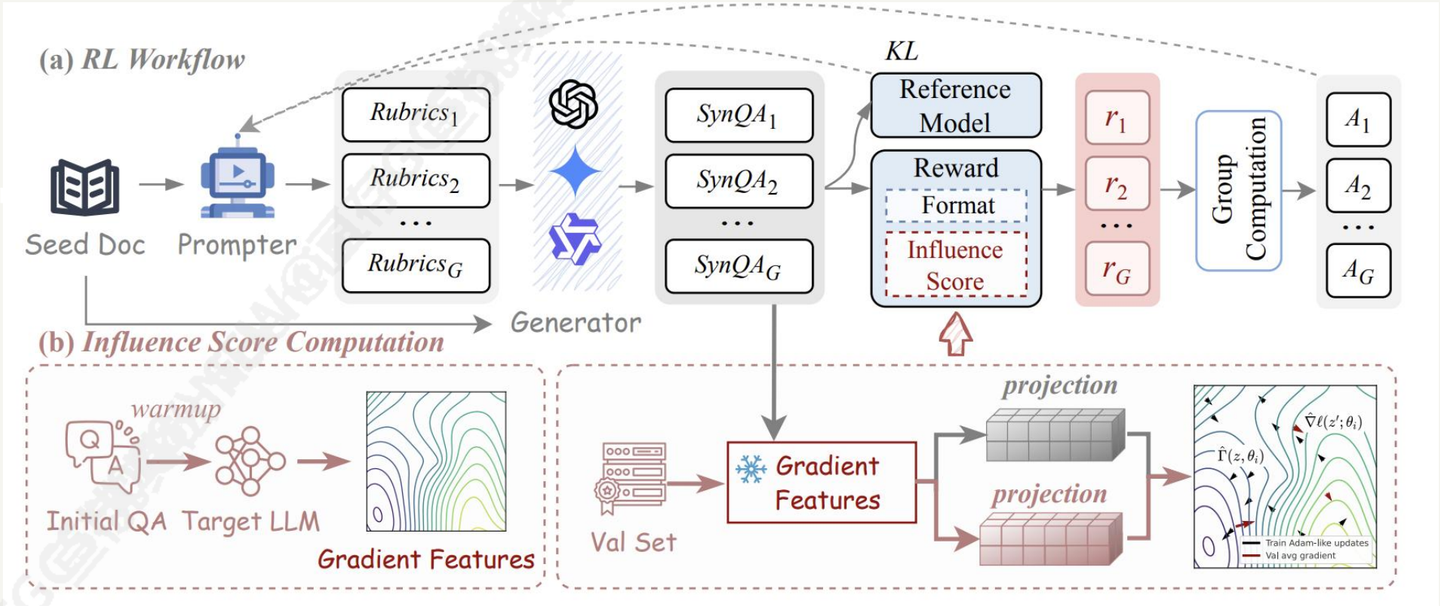
直接衡量"训练这个 task 是否让 validation 变好"



- Signal 直接刻画训练效果是否对齐下游目标
- 完全绕开了 difficulty 的 proxy

OptimSyn — 闭环到生成

Selection 内化到 Generation 中，"造"和"选"统一



- 用 gradient-based influence 作为 reward 信号
- 通过 **GRPO** 训练 rubric generator

Socratic-SWE 的 Generator Reward

$$R_G = \text{Valid}(\tau, v, r) \cdot \cos(g_\tau, G_v)$$

Valid：四层执行验证

- **Format**：格式正确性
- **Grounding**：上下文关联
- **Execution**：可执行性
- **Semantics**：语义正确性

cos：梯度方向对齐

Cosine 隔离**方向信号**

Inner product 会被 gradient magnitude confound

Multi-file 大 patch 天然梯度大，但不一定有用

实验对比

Hardness: **47.4%**　　Variance: **48.8%**　　**Gradient-aligned: 50.4%**

Cosine 50.40% vs Inner product 49.40% —— 方向信号比幅度信号更可靠

Insight：纯 gradient 对齐已隐式编码合适难度，无需额外 difficulty prior（hybrid 消融 +0.20）

Act 2 小结

奖励对齐的演进

关键洞察

Evaluation signal **必须回流到 generation** 中，形成闭环

Task 的"有用性"不是 task 本身的属性，而是 **task 和 model current state 的关系**

03

ACT 3

环境构造

让 Loop 跑得起来

环境交互成本是 scaling 的 **最后瓶颈**

路线一：World Model 替代环境

WebWorld

1M+ 真实交互轨迹训练 web world model，支持 30+ 步 long-horizon simulation

SearchGym

合成知识图谱和对齐语料，做 search agent 的无成本训练环境

AWM (Agent World Model)

Code-driven + database-backed 路线，1000 个合成环境，比 LLM 模拟更稳定

Qwen-AgentWorld

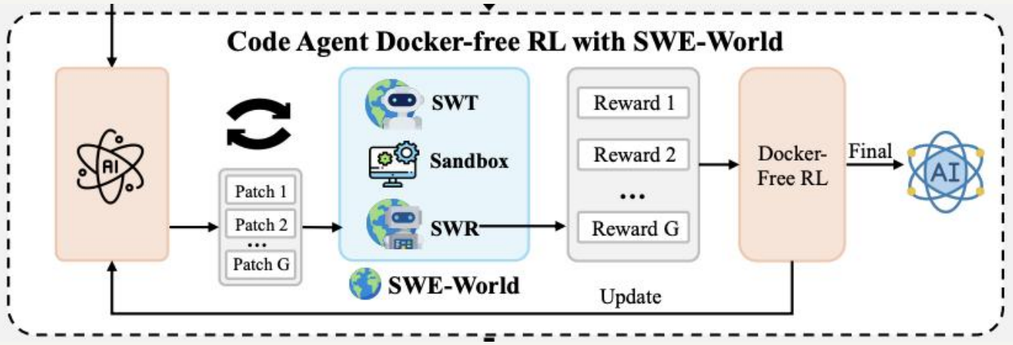
原生 Language World Model，7 个域统一（MCP/Search/Terminal/SWE/Android/Web/OS）

SWE-World (SWE 落地)

LLM 做 Transition Model + Reward Model 替代 Docker 执行，全 Docker-free 的 SFT 和 RL pipeline

性能：**6.2% → 68.2%**

优势：速度快、成本低　　**劣势**：模拟终究不是真实执行，fidelity 有限



路线二（主推）：让真实执行更便宜

SWE-MiniSandbox

核心洞察：隔离不需要完整 Docker

- Linux namespace + chroot 做 kernel-level 隔离
- Python venv 替代完整容器镜像

Storage: **5%**　Setup: **25%**

性能与 Docker pipeline **完全一致**

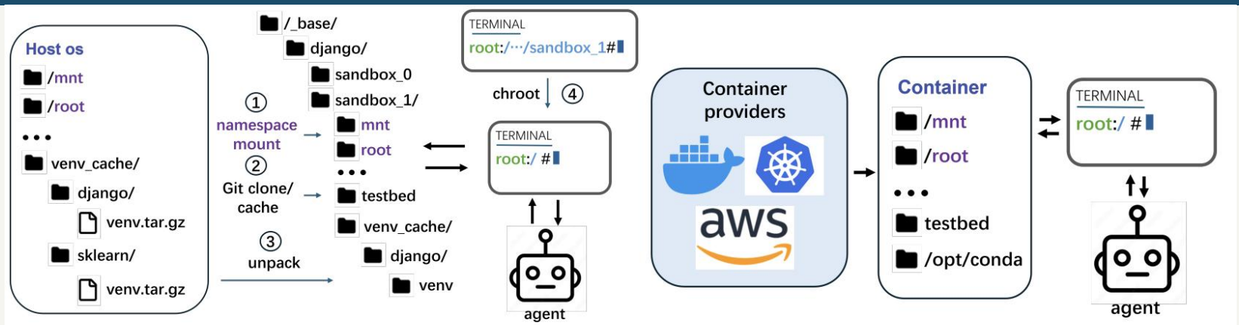
SWE-Hub

环境构造自动化

- Env Agent：raw repo → 可复现执行环境
- Bug Agent：大规模合成 task

→ Production-grade **"数据工厂"**

真实执行反馈的 fidelity 不可替代



(a) SWE MiniSandbox　　*(b)* Container

两条路线互补

World Model

- Fast exploration
- Cheap filtering（初筛）

角色：Pre-filter

+

Real Environment

- Grounding + Verification
- Reward computation（终判）

角色：Grounding

正确做法：World model 做 pre-filter，real execution 做 grounding

两条路线不是对立，而是互补

展望

01

打破 Closed-World

当前 skill registry 在固定 repo pool 上会饱和

→ 动态扩展 repo 池是持续进化的关键

02

Cross-Domain Transfer

SWE skill → Terminal / DevOps 任务

初步验证：Terminal-Bench 2.0 **+4.50**

03

环境生成作为 Learnable Policy

Env Agent 也能被 RL 优化

→ 真正实现"环境的生成、验证与持续演化"

让训练 Agent 的系统本身变得 Agentic

THANK YOU

Thank You

Q & A

焦政博

香港中文大学 MMLab / 腾讯混元青云计划

论文：arXiv:2606.07412

主页：Frostlinx.github.io